PCT/GB ~~99~~/ 03100

17 SEPTEMBER 1999

The Patent Office

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South ~~Wales~~
NP10 8QQ

ESU

REC'D 23 NOV 1999
WIPO PCT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

I also certify that the attached copy of the request for grant of a Patent (Form 1/77) bears an amendment, effected by this office, following a request by the applicant and agreed to by the Comptroller-General.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Signed [illegible]

Dated 29 OCT 1999

Patents Form 1/77

s Act 1977
6)

The Patent Office

1/77

13OCT98 E396672-1 C45919
P01/7700 0.00 - 9822191.4

THE PATENT OFFICE
J
13 OCT 1998
RECEIVED BY POST

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference — CRT/DCT/121098

2. Patent application number *(The Patent Office will fill in this part)* — **9822191.4** — 13 OCT 1998

3. Full name, address and postcode of the or of each applicant *(underline all surnames)* — MACIEJ KUBICZEK, CHRISTOPHER ROBERT TURNER
   BOTH OF: 3 WESTBROOK AVENUE
   HAMPTON
   MIDDLESEX
   TW12 2RE

   Patents ADP number *(if you know it)*

   If the applicant is a corporate body, give the country/state of its incorporation

   7531148001 6530596001

4. Title of the invention — HIGH PERFORMANCE LOW COST MICROPROCESSOR

   6

5. Name of your agent *(if you have one)* — ~~NONE~~

   "Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

   ~~C. R. TURNER~~
   ~~3 WESTBROOK AVENUE~~
   ~~HAMPTON~~
   ~~MIDDLESEX~~
   ~~TW12 2RE~~

   MARKS & CLERK
   4220 NASH COURT
   OXFORD BUSINESS PARK SOUTH
   OXFORD
   OX4 2RU

   5/PA
   14/4/99

   Patents ADP number *(if you know it)*

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*
/		

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*
/	

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*
   a) any applicant named in part 3 is not an inventor, or
   b) there is an inventor who is not named as an applicant, or
   c) any named applicant is a corporate body.

   NO

9. **Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document**

Continuation sheets of this form	Ø
Description	6
Claim(s)	1
Abstract	1
Drawing(s)	5 ~~[illegible]~~

8

10. **If you are also filing any of the following, state how many against each item.**

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. ~~I~~/We request the grant of a patent on the basis of this application.

Signature [illegible] Date 12-10/98

12. Name and daytime telephone number of person to contact in the United Kingdom C. R. TURNER ~~0181 624 82~~ 0181 408 5266

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

Background to the Invention

1. Field of the invention

The present invention relates to a simplified instruction set microprocessor (RISC). In particular, a microprocessor tailored toward and capable of implementing bytecoded virtual machines at speeds comparable to existing language-specific microprocessors, but an order of magnitude smaller in terms of the number of transistors necessary to implement the microprocessor compared to existing designs, such as Java chips. In a preferred embodiment, the microprocessor will be used to implement a version of the Java Virtual Machine. The small size and resulting low-power consumption make the microprocessor especially ~~attractive for use in embedded applications.~~

2. Description of prior art

High level language oriented computer architectures have been known for many years, but until now few of them have been successful commercially. Currently, the most popular computer architectures are either general-purpose Complex Instruction Set Computers (CISC) such as the Intel 80x86 family of microprocessors, or general-purpose Reduced Instruction Set Computers (RISC).

The success of the Java programming language, and the demands, which the Java environment places on an execution platform, can change this situation. This means that language specific architectures may become popular, since a Java-oriented architecture is bound to be more efficient in executing Java than a general-purpose microprocessor.

Java is a young technology and has been designed to run on a variety of computing platforms, but the benefits of using dedicated special-purpose architectures have been apparent from the start. SUN Microsystems, the originator of Java, have developed a microprocessor core called picoJava. This core is targeted toward the high-end of the computer market, i.e. it is envisaged to be used in large desktop computer type applications. It is large, complex and power-hungry, making it unsuitable for smaller, embedded type of applications. Certain similarities between the Java Virtual Machine and the Forth programming language have led manufacturers of Forth chips to re-brand them as Java chips. While some Forth chips are better at implementing Java than other general-purpose microprocessors, differences between the Java Virtual Machine and Forth ensure that a dedicated Java microprocessor will outperform a Forth microprocessor.

Embedded systems often operate under stringent real-time constraints. In particular, the interrupt latency of a processor is a critical parameter. With complex microcoded instructions, interrupt latency can be on the order of several dozen or even hundreds of clock cycles.

Summary of the Invention

It is an object of this invention to provide a high-performance, virtual machine tailored microprocessor with a reduced transistor count compared to other processors, such as existing Java processors.

It is also an object of this invention to provide a high-performance microprocessor, which has low interrupt latency necessary in many real-time embedded systems.

It is a further object of the invention to provide extremely good code density for high-level language generated programs.

It is a further object of the invention to simplify the translation process from a machine independent bytecode representation of a program, such as the Java Virtual Machine bytecodes, to the microprocessor's instruction set, so that "on the fly" application program loaders, such as the Java class loaders can be implemented at minimal cost.

It is a further object of the invention to provide a means for the microprocessor to communicate with other microprocessors via a local area network, and to enable software upgrades to be performed remotely over the network.

It is a further object of the invention to provide the means for the microprocessor to interface directly to a dedicated slave co-processor (such as a numeric co-processor, a digital signal processor (DSP), a special purpose communication processor, etc.) so that special purpose systems can be easily implemented.

The above and related objectives may be achieved by the use of the novel, low cost microprocessor herein disclosed. In accordance with one aspect of the invention, a microprocessor system in accordance with this invention has a central processing unit, an 8-bit wide bytecode memory, a 32-bit data memory and busses connecting the central processing unit with the two memories.

In accordance with another aspect of the invention, the microprocessor system contains an arithmetic logic unit and a data stack. The top two elements of the data stack are connected to the inputs of the arithmetic logic unit and the output of the arithmetic logic unit is connected to an internal data bus.

In accordance with yet another aspect of the invention, the top three elements of the data stack contain special-purpose circuits, which enable the efficient (single cycle) execution of seven primitive stack operations directly in hardware. The remaining data stack elements are simple shift registers.

In accordance with another aspect of the invention, the microprocessor has a means for distinguishing a "fast call" instruction from a regular instruction by utilising circuitry, which decodes a particular bit of the 8-bit bytecode. In a preferred embodiment of the microprocessor, this would be the most significant bit of the 8-bit bytecode.

In accordance with another aspect of the invention, the microprocessor has a means of recognising a "fast return" instruction "folded" with a regular instruction, by utilising circuitry, which decodes the "fast call" bit in the 8-bit bytecode and another dedicated bit in the 8-bit bytecode. In a preferred embodiment of the microprocessor, this would be the second most significant bit in the 8-bit bytecode.

In accordance with another aspect of the invention, the microprocessor contains a dedicated register called the *Parameter Pool Pointer*, together with associated circuitry and several dedicated instructions for storing and accessing 32-bit quantities in data memory using short (8-bit) offsets. This mechanism allows the efficient implementation of local (dynamic) variables in block and object oriented languages.

In accordance with another aspect of the invention, the microprocessor contains a dedicated register called the *Global Constant Pool Pointer*, together with associated circuitry and dedicated instructions for storing and accessing 32-bit quantities in data memory using short (8-bit) offsets. This mechanism allows the efficient implementation of 32-bit literal constants, which are global to all execution contexts, using only an 8-bit bytecode extension.

In accordance with another aspect of the invention, the microprocessor contains a dedicated register called the *Local Constant Pool Pointer*, together with associated circuitry and dedicated instructions for storing and accessing 32-bit quantities in data memory using short (8-bit) offsets. This allows the efficient implementation of 32-bit literal constants, which are local to a particular execution context.

In accordance with another aspect of the invention, the microprocessor contains a dedicated register called the *Extension Stack Pointer*, together with dedicated circuits, which is used to spill data stack elements into data memory, and to refill the data stack from data memory.

In accordance with another aspect of the invention, the microprocessor contains dedicated circuitry, to efficiently implement a subroutine call via an in-memory jump table, which is essential for an efficient implementation of dynamic method dispatch in object oriented programming languages. In a preferred embodiment of this invention this language would be the Java programming language.

In accordance with another aspect of the invention, the microprocessor contains dedicated circuitry and instruction to improve the efficiency of exception handlers.

In accordance with another aspect of the invention, the microprocessor contains dedicated hardware mechanisms to allow the efficient implementation of a variety of garbage-collection algorithms, which are essential in many object-oriented systems, such as Java.

In accordance with yet another aspect of the invention, the microprocessor contains circuits for interfacing the microprocessor to a data network and to allow the microprocessor's software to be dynamically upgraded over that network.

In accordance with another aspect of the invention, the microprocessor contains a means for communicating with a special-purpose co-processor, such as a DSP processor or a math processor. The co-processor can be integrated on the same silicon die as the microprocessor, or can be located off-chip.

The attainment of the foregoing and related objects, advantages and features of the invention should be more apparent to those skilled in the art, after review of the following more detailed description of the invention.

Detailed Description of the Invention

The microprocessor of this invention has the following important distinguishing features:

- Low transistor count and low power dissipation for use in small embedded systems
- Unique two-level architecture involving a fast *micro machine* and a slower *macro machine*
- Modified Harvard architecture with an 8-bit wide *bytecode memory* and a 32-bit wide *data memory.*
- Bytecode (0-operand) instruction set for maximum code density.
- 32-bit internal architecture.
- A 32-bit wide hardware operand stack coupled to the ALU, with automatic fill/spill unit.
- A 32-bit wide hardware return (subroutine) stack.
- RISC-like instruction set, with most instructions executing in a single cycle.
- No pipelining.
- Zero-overhead subroutine return instruction which can be "folded" with other instructions.
- 128 user-programmable bytecodes facilitating the efficient creation of virtual machines.
- Hardware support for the efficient execution of high-level languages, such as Java.
- Global and local constant pools, allowing the specification of 32-bit immediate constants using 8-bit offsets in the bytecode stream.
- Parameter pool, allowing the efficient implementation of local variables.
- Hardware and instructions for dynamic method dispatch
- Hardware and instructions for the efficient implementation of software exceptions
- Hardware support for garbage collection algorithms
- Interface to a local area network for dynamic upgrading of the application software
- Hardware interface to a variety of on- and off-chip co-processors

Figure 1 shows the block diagram of the microprocessor **(10)**. It is seen that the microprocessor has a simple architecture. The central processing unit is connected to two memories: an 8-bit wide Bytecode Memory **(100)** and a 32-bit wide Data Memory **(114)**. The CPU contains a main Arithmetic Logic Unit **(103)** and two hardware stacks: the Return Stack **(104)** and the Data Stack **(108)**. The top two elements of the Data Stack **(108)** are connected directly to the inputs of the ALU **(103)**. The CPU also contains an 8-bit Instruction Register **(101)** for latching the currently executing instruction bytecode, and an Immediate

Operand circuit (**102**) which connects the output of the Bytecode Memory to the CPU's Internal Data Bus (**115**). The CPU also contains a Program Counter register (**105**), which is connected to the input of the Bytecode Memory Address ALU (**106**) and also to the input and output ports of the Return Stack (**104**). In addition to the above, the CPU also contains four dedicated address registers, namely the Global Constant Pool Pointer (**109**), the Local Constant Pool Pointer (**110**), the Extension Stack Pointer (**111**) and the Parameter Pool Pointer (**112**). The read ports of the address registers are connected to the input of the Data Memory Address ALU (**113**).

Figure 2 shows the block diagram of the fast subroutine call mechanism. The microprocessor instruction decode logic includes a circuit which distinguishes a bit (**150**) (the most significant bit in a preferred embodiment of the invention) in the 8-bit instruction register (**101**) which latches a bytecode fetched from bytecode memory (**100**). If the bit (**150**) is active, the microprocessor program counter is loaded with the output of the Address Generator circuit (**151**) which uses the remaining 7 bits of the bytecode to produce an address. In a preferred embodiment of the invention, the circuit (**151**) shifts the low 7 bits left by two bits. The program counter register load is determined by the control signal on gate (**152**). At the same time, the current value of the program counter (**105**) is stored in the on-chip return stack (**104**). This sequence of actions is performed in a single clock cycle, and is equivalent to a fast subroutine call to one of 128 possible locations.

Figure 3 shows the block diagram of the circuit implementing the folding of a subroutine return operation with a regular instruction. The microprocessor instruction decode logic includes a circuit which tests two bits (**150** and **200**) in the 8-bit instruction register (**101**) which latches a bytecode fetched from bytecode memory, and controls a circuit (**201**), which reloads the program counter register from the top of the return stack. In a preferred embodiment of the invention the two bits would be the two most significant bits in the 8-bit bytecode This action is performed in parallel with normal instruction execution, and means that any (regular) instruction of the microprocessor can be "folded" with a return from subroutine operation, effectively providing a zero-overhead operation.

The fast call/fast return features of the microprocessor allow very short instruction sequences to be encoded as 8-bit user-defined bytecodes. This feature is a key to providing good code density and also good interrupt latency, since the "macro" instructions are composed of a sequence of simple (RISC-like) machine instructions of the microprocessor, and can be interrupted without problems.

The instruction decode logic of the microprocessor includes a circuit, shown in Figure 4, which allows a bytecode to be followed by a single 8-bit immediate value. This immediate value is read from bytecode memory (**100**) and latched in the immediate operand module (**102**). Depending on the bytecode, this 8-bit value can be combined with one of the address registers (**109, 110, 111, 112**), shown in Figure 4 as (**250**), to provide an address into data memory, from which a full 32-bit immediate value is fetched.

The organisation of the Data Stack is shown in Figure 5. The top three data stack entries (**300**) are different ~~from the remaining stack entries (**301**) and are provided with special-purpose circuits, which enable them to~~ execute seven primitive stack manipulation operations directly in hardware in one clock cycle. The top two elements of the Data Stack are connected to the inputs of the microprocessor's Arithmetic Logic Unit (**103**). The stack manipulation primitives have been selected to either directly correspond to some Java Virtual Machine stack manipulation instructions, and to allow the composition of the remaining Java Virtual Machine instructions from two or three primitives. The primitive operations are:

• POP	Remove the top stack element
• DUP	Duplicate the top stack element
• OVER	Copy the second stack element over the top stack element
• SWAP	Swap the top two stack elements
• LROT	Rotate the top 3 stack elements left
• RROT	Rotate the top 3 stack elements right
• TOVER	Copy the third stack element over the top stack element

The remaining Data Stack elements (**301**) are implemented as a simple array of 32xn shift registers.

Object-oriented languages, such as Java, require the efficient implementation of local (or dynamic) variables. The microprocessor here described supports the concept of a *Parameter Pool*. A parameter pool is an area of data (32-bit) memory, with individually addressable locations. The addresses of the individual locations are small positive integers. The Parameter Pool is implemented using a dedicated pointer register called the *Parameter Pool Pointer* register. Hardware mechanisms are provided for transferring data from the data stack to the Parameter Pool, and for accessing and storing individual elements in the pool using either a short (8-bit) offset in the bytecode stream, or a 32-bit offset from the top of the data stack.

A 32-bit architecture requires 32-bit literal constant operands to be included in the instruction set. Current practice in bytecoded architectures is to embed the literal constant in the bytecode stream. This increases the code size, and makes the instruction decoding circuits more complex. The approach taken with the present microprocessor is the provision of *constant pools*, which hold the values of the literal constants. Two pools are provided for each execution context. A *Global Constant Pool* contains literal constants common to all execution contexts (the most commonly occurring constants, plus constants for the operation of the virtual machine). A *Local Constant Pool* contains literal constants specific to a particular execution context. The constant pools are implemented using two dedicated pointer registers: *the Global Constant Pool Pointer* register and the *Local Constant Pool Pointer* register. Hardware mechanisms are provided to enable the initialisation of the constant pools and the efficient retrieval of any constant from the pools using an 8-bit offset in the bytecode stream, or a 32-bit offset from the top of data stack.

The on-chip data stack is used for expression evaluation. In a preferred embodiment of the invention the depth is equal to 8, which is adequate for most situations. The (relatively) shallow depth of the data stack makes context switching faster, since in the case in question only at most 8 elements need to be spilled into memory. In some cases, the number of elements on the data stack may exceed 8, causing stack overflow. To deal with this situation a dedicated register, called the *Extension Stack Pointer* is provided, together with dedicated circuits for loading/storing the bottom element of the data stack in data memory, according to the address stored in the Extension Stack Pointer.

Object oriented languages use dynamic method dispatch very extensively. The present microprocessor implements dynamic method dispatch using subroutine calls via a jump table. This operation is performed using a built-in instruction of the microprocessor.

Many modern programming languages, such as Java, include facilities for defining exception handlers. The present microprocessor has instructions and dedicated circuits to improve the efficiency of the implementation of exception handlers.

Modern high-level programming languages, such as Java, use sophisticated memory management techniques based on garbage collection. The present microprocessor contains circuits, which allow the efficient implementation of a variety of garbage collection algorithms, for different application areas.

Since the "semantic gap" between the microprocessor here described and typical stack-based virtual machine architectures is small, it is possible to implement very simple dynamic loaders, which allow machine-independent application code (such as software upgrades) for the microprocessor to be downloaded over a local data network. In a preferred embodiment of the invention, the machine-independent code would be the Java class file format. For this reason, the microprocessor contains a unit for connecting the processor to a local area network.

Many modern embedded systems consist of a control module and a data-processing module. The control module is usually implemented by a general-purpose microprocessor, while the data processing part is implemented by a dedicated hardware processor. Depending on the application area, the dedicated hardware processor can be a digital signal processing (DSP) processor, or a special-purpose math co-processor. To allow the present microprocessor to be used in such situations, special purpose instructions have been provided in the microprocessor, together with dedicated circuitry enabling the processor to directly interface to a wide variety of special-purpose co-processors.

CLAIMS

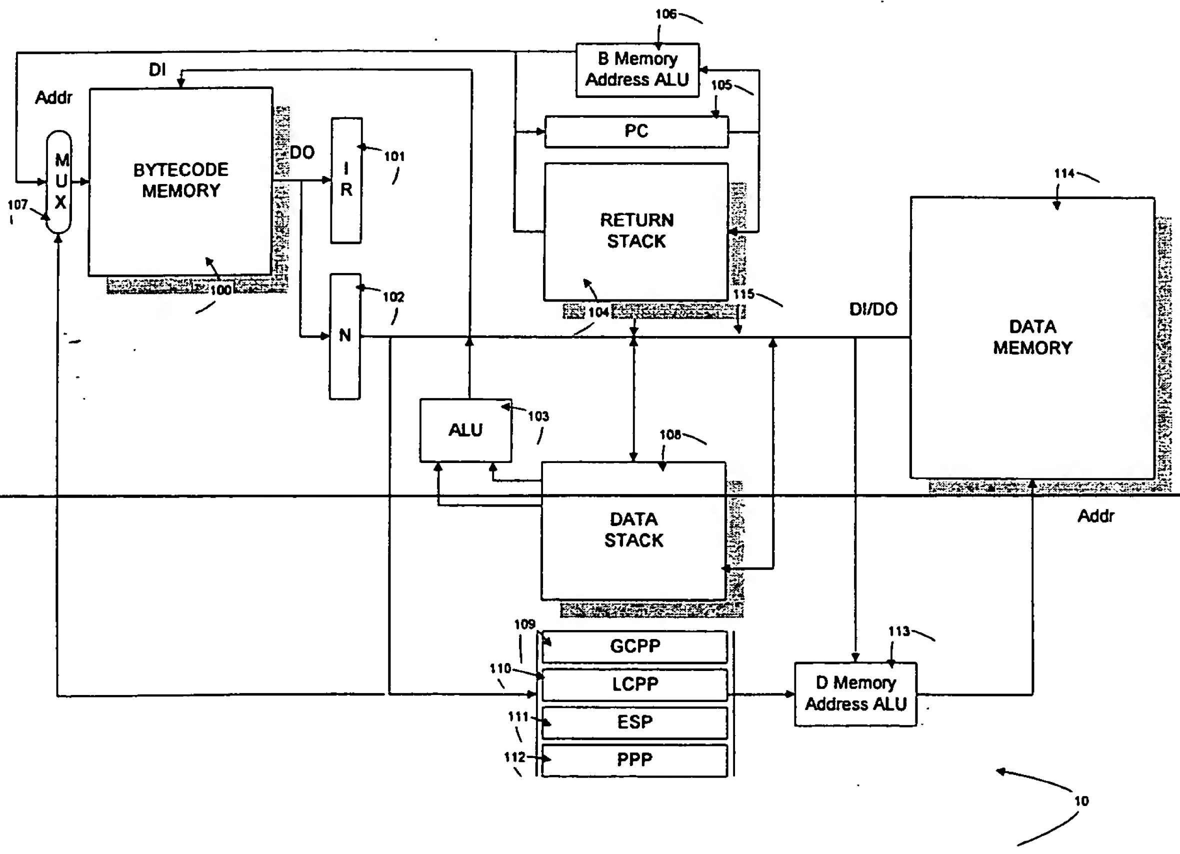
What is claimed is:

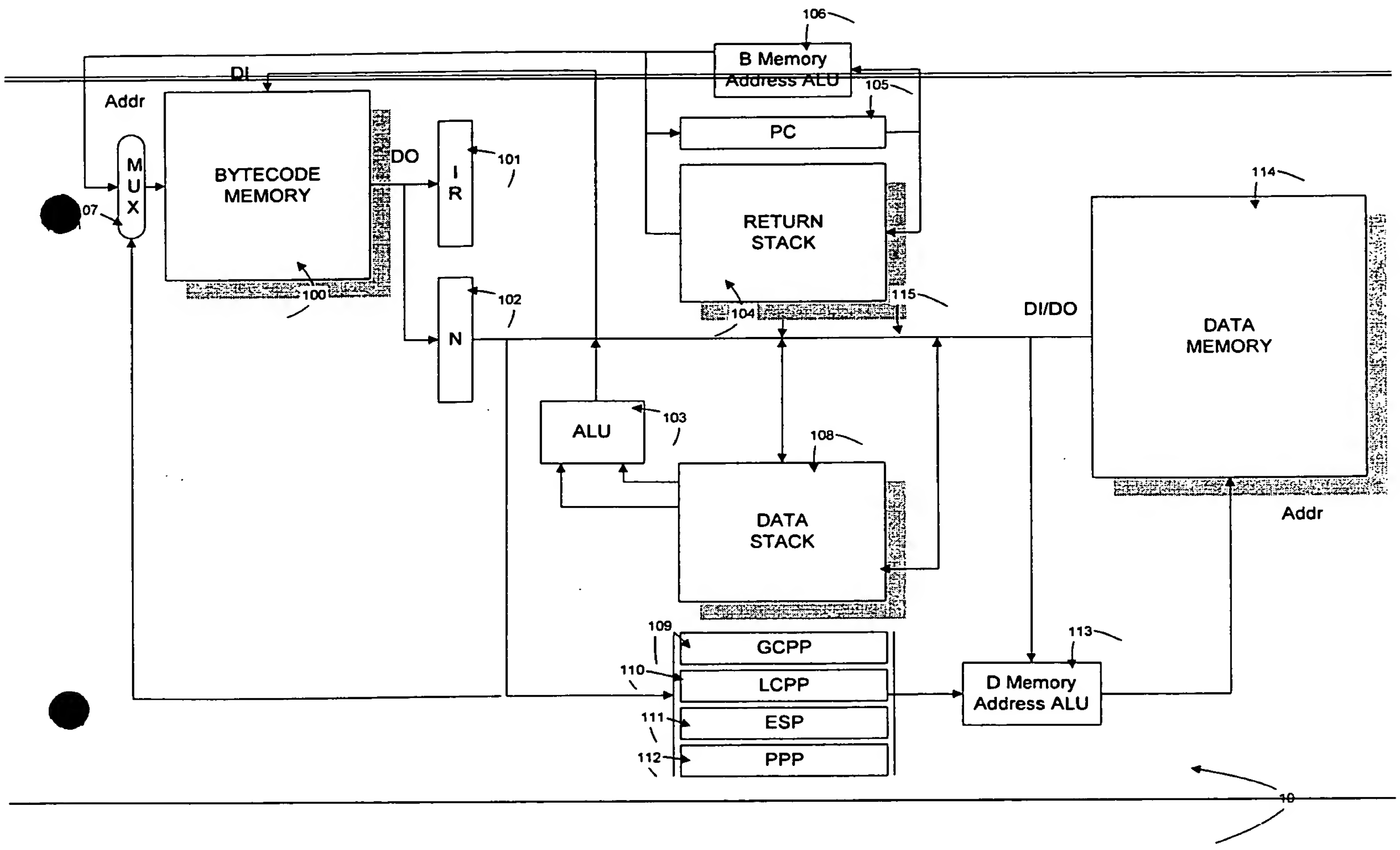
1. A microprocessor system, consisting of a central processing unit, 8-bit wide instruction memory, 32-bit wide data memory and a hardware stack connected via an internal bus to the program counter register, together with an instruction decode unit which includes a circuit for detecting the presence of a distinguished bit in the 8-bit bytecode, together with a circuit for loading the remaining bits of the bytecode shifted left by a number of bits into the microprocessor's program counter register, while at the same time storing the current value of the program counter register on the aforementioned stack.
2. The microprocessor system of claim 1, additionally comprising a circuit in the instruction decode logic section which tests two distinguished bits, one of which is the bit described in claim 1, in an 8-bit ~~bytecode, controlling a circuit which loads the program counter register with a value from the stack~~ mentioned in claim 1.
3. The microprocessor system of claim 2, additionally comprising two address registers, additional circuitry and instructions allowing the fetch of 32-bit immediate constants using an 8-bit offset embedded in the bytecode instruction stream.
4. The microprocessor system of claim 3, additionally comprising an address register, additional circuitry and instructions allowing the fetch and store of 32-bit values using an 8-bit offset embedded in the bytecode instruction stream.
5. The microprocessor system of claim 4, additionally comprising an instruction for the efficient execution of a subroutine calls via an in-memory jump table.
6. The microprocessor system of claim 5, additionally comprising an instruction and dedicated hardware circuits for the efficient execution of structured exception handlers of modern high-level languages.
7. The microprocessor system of claim 6, additionally comprising instructions and dedicated hardware for the efficient implementation of a variety of garbage-collection algorithms.
8. The microprocessor system of claim 7, additionally comprising instructions a hardware interface to a local area network.
9. The microprocessor system of claim 8, additionally comprising circuits and instructions for interfacing with on- or off-chip co-processors such as numeric or digital signal processing co-processors.
10. The microprocessor system of claim 3, additionally comprising of a 32-bit wide data stack, whose top three elements are provided with a hardware means for executing seven distinct operations on the said three elements and whose remaining elements are implemented as an array of simple shift registers.

Abstract

High Performance Low Cost Microprocessor

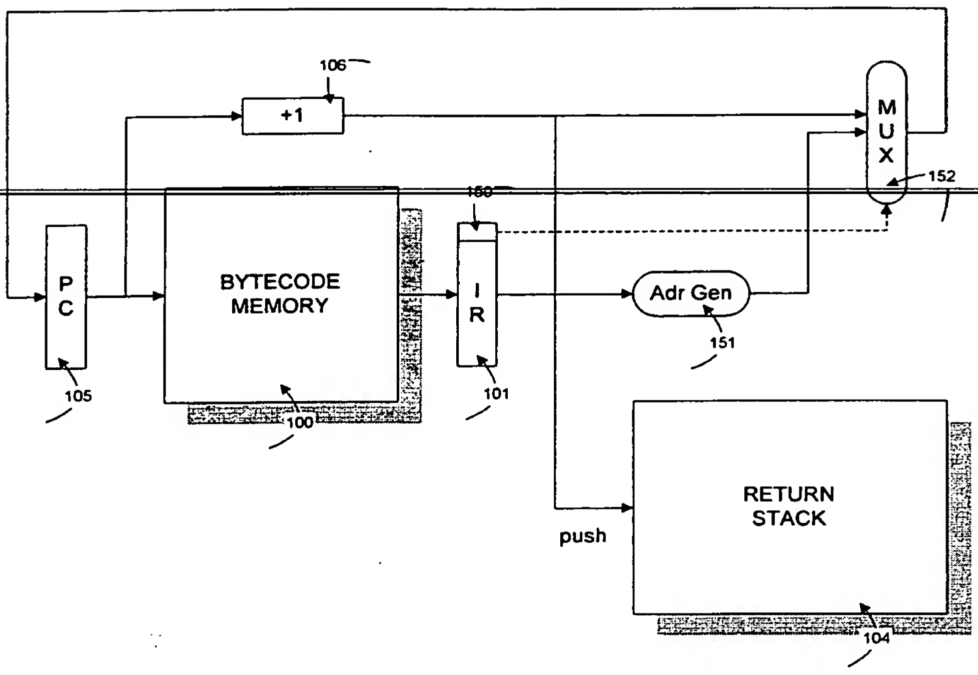
A microprocessor (**10**) incorporating a modified Harvard architecture connected to two memory banks (**100**) and (**114**). The main CPU also includes two pushdown stacks (**104**) and (**108**). One of the stacks has its top two items connected directly to an arithmetic-logic unit (**103**). In addition hardware is provided to perform seven operations on the top three stack elements. The instruction length of the microprocessor is 8 bits and most instructions execute in a single clock cycle. A unique feature of the instruction set is that 128 of the 256 possible bytecodes are user-programmable. Each regular instruction can be "folded" with the "return from subroutine" instruction. This allows the efficient implementation of a wide variety of virtual machines, such as the Java Virtual Machine. The microprocessor also contains dedicated registers and circuits for the efficient implementation of dynamic variables (**112**), 32-bit immediate constants (**109**) and (**110**), interfaces to dedicated co-processors and interfaces to local area networks allowing dynamic upgrades of application software.





106
B Memory Address ALU
105
DI
Addr
BYTECODE MEMORY
DO
I R
101
M U X
07
100
N
102
PC
RETURN STACK
104
115
DI/DO
114
DATA MEMORY
Addr
ALU
103
108
DATA STACK
109
GCPP
110
LCPP
111
ESP
112
PPP
113
D Memory Address ALU
10

Figure 1



106
+1
M
U
X
152
150
P
C
105
BYTECODE
MEMORY
100
I
R
101
Adr Gen
151
push
RETURN
STACK
104

Figure 2

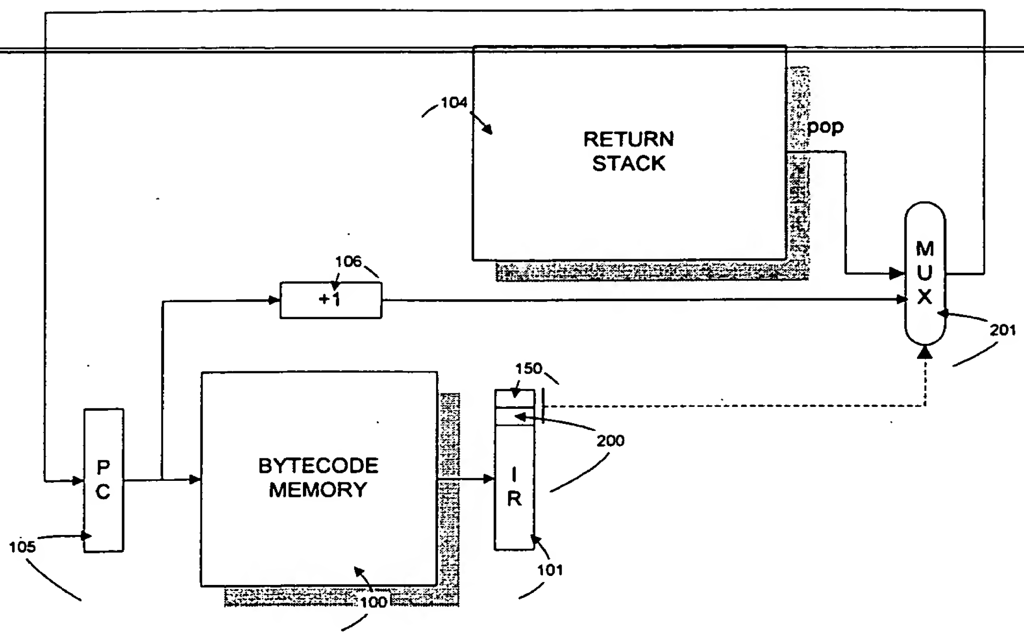
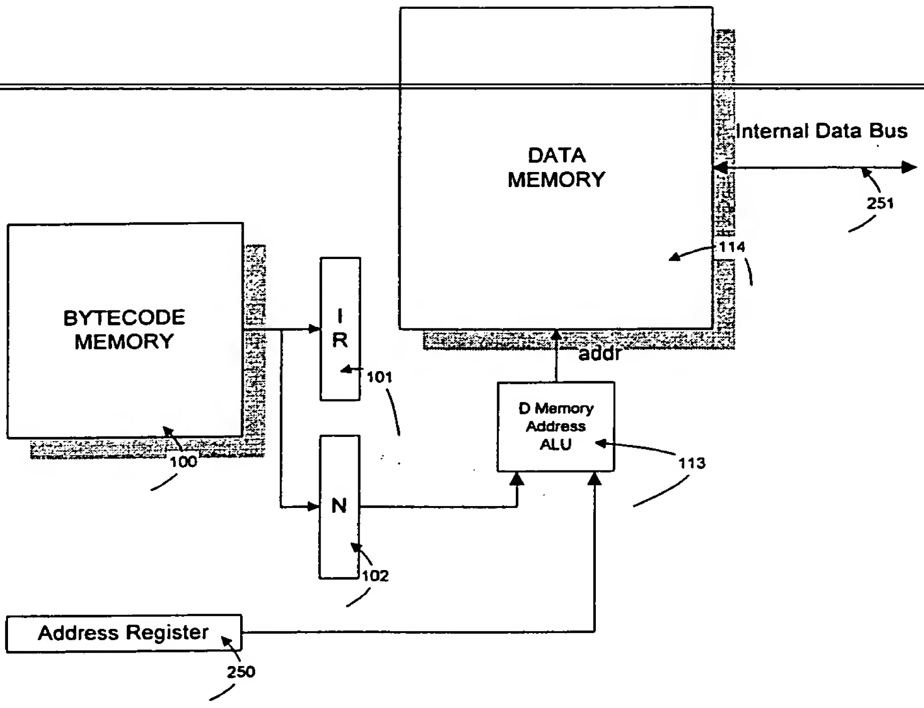

104
RETURN STACK
pop
106
+1
MUX
201
150
200
P C
105
BYTECODE MEMORY
100
I R
101

Figure 3



DATA MEMORY
Internal Data Bus
251
114
BYTECODE MEMORY
I R
101
addr
D Memory Address ALU
113
100
N
102
Address Register
250

Figure 4

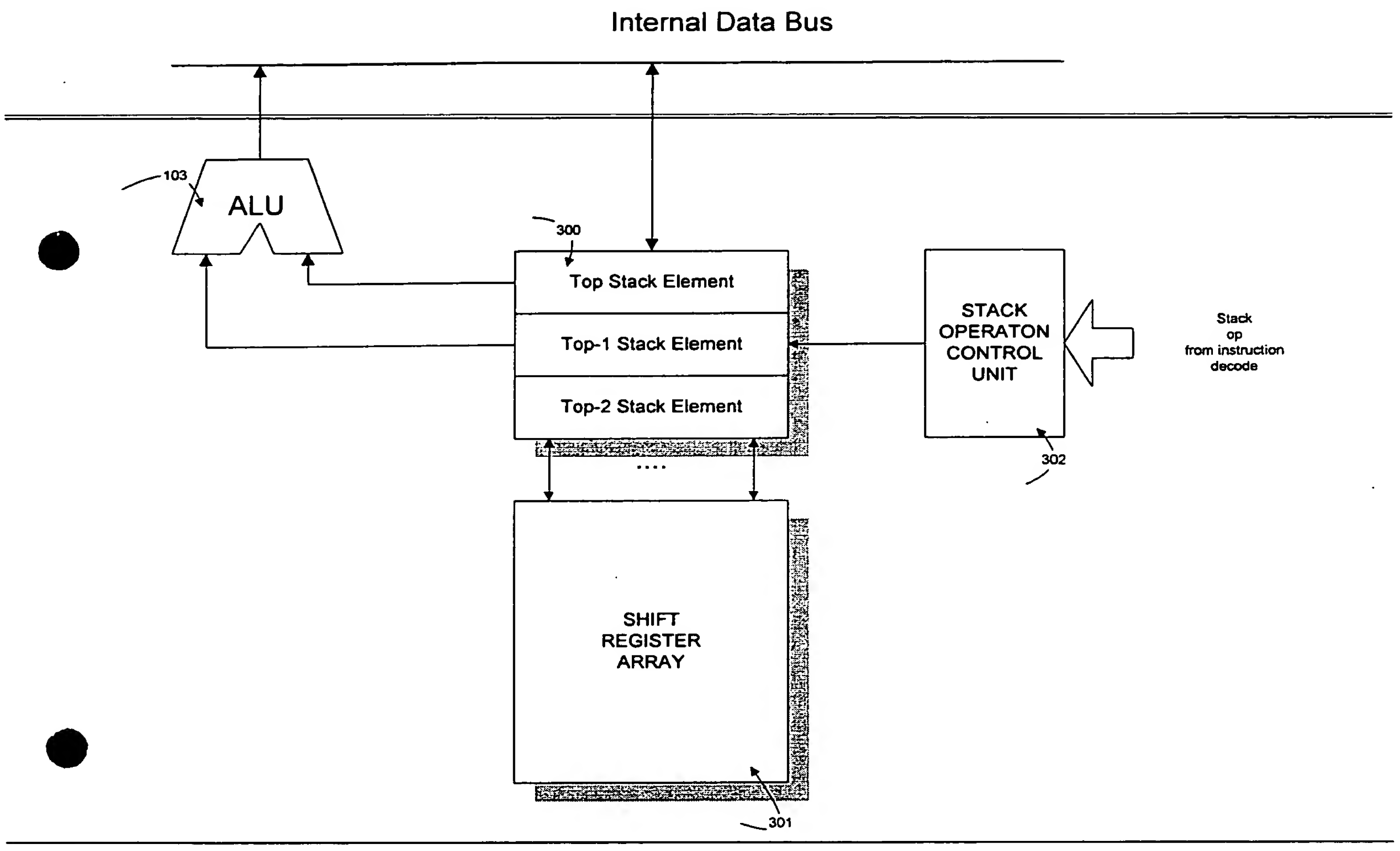

Internal Data Bus
103
ALU
300
Top Stack Element
Top-1 Stack Element
Top-2 Stack Element
STACK OPERATON CONTROL UNIT
Stack op from instruction decode
302
....
SHIFT REGISTER ARRAY
301

Figure 5